

## RESEARCH PROPOSAL

# Towards a Generalized Framework for Secure Time-Stamping

Keno Goertz

February 24, 2025

## 1 Problem statement

Many applications require associating a moment in time with digital data: Be it the time of financial transactions in a payment system or the date of a patent's registration.

Digital time-stamps make this association between data and time verifiable: A time-stamp provides some way for a third party to verify that the datum was actually in existence at the time associated with it.

Many protocols for digital time-stamping have been proposed over the 90's and 2000's. What started with the simple idea of letting a trusted third party issue time-stamps, eventually developed into the modern notion of a blockchain. Section 2 provides an overview of time-stamp protocols published in the literature.

As the example of blockchains shows, time-stamp protocols from the 90's can be used as building blocks for modern and practical distributed protocols, with applications going far beyond the original idea of attaching a time to a datum. Yet, the relationship between the different suggested time-stamp protocols seems to be poorly understood.

I think it likely that important insights into distributed system design can be gained by figuring out how the different proposed time-stamp protocols could be unified into one generalized time-stamping framework. The framework would provide a clear picture of how the building blocks used within time-stamp protocols relate to each other. This may help in identifying trade-offs between different approaches to time-stamping. It

would also facilitate the adoption of a time-stamp protocol for a specific use case: Rather than having to build a new protocol from scratch, the time-stamping framework could be consulted to construct a protocol tailored to the requirements of the use case.

In my thesis, I will unify the existing approaches to time-stamping in such a generalized framework. I will go on to identify potential trade-offs concerning time-stamp resolution, security and performance. Finally, I will conduct experiments on some of the identified trade-offs using a practical implementation of a time-stamp protocol.

## 2 Related Work

**Trusted Time-Stamping** Perhaps the simplest time-stamp protocol is based on central trust in a Time-Stamp Authority (TSA), acting as a trusted third party for issuing time-stamps. In such a scheme, a client sends the cryptographic hash of the data for which it desires a time-stamp to the TSA. The TSA then issues a time-stamp by signing the client-provided hash along with the current time using public-key cryptography. [4] The security of trusted time-stamping depends on complete trust in the time-stamp authority and the security of its private key. The time-stamp protocols described in the following paragraphs reduce the level of trust that needs to be placed in a single entity.

**Random-Witness Time-Stamping** To decrease the trust required in a single TSA, it can instead be distributed among witnesses selected from a fixed set of  $N$  servers. [4] Each of the servers is assigned a unique ID. A valid time-stamp requires the signature of  $k$  servers, which are selected by interpreting the outputs of a pseudo-random number generator (PRNG) as a set of server IDs. The PRNG is seeded with the hash of the data to be time-stamped.

An attacker who wants to forge a back-dated time-stamp would have to control all  $k$  servers which were selected as random witnesses by the PRNG. Even assuming the attacker had control over more than  $k$  servers, this is still very unlikely to happen.

**Threshold Cryptography** Threshold cryptography simplifies the public key infrastructure of a distributed time-stamping scheme. [6] A private key is shared among servers participating in the time-stamping scheme in such a way that the collaboration of  $k$  servers is required in order to produce a valid signature for a time-stamp certificate. Clients then only need to maintain a single public key. However, the security of this scheme depends completely on the attacker controlling less than  $k$  nodes.

**Linked Time-Stamping** Another way to establish trust in a TSA is to have it chain time-stamps together. [4] In this approach, each time-stamp contains information on the time-stamp preceding it. Time-stamps of the chain need to be regularly and widely publicized (e. g. by printing them in a newspaper). A time-stamp can then be verified by ensuring that it fits into the publicized chain. The existence of a publicly known chain prevents the TSA from back-dating time-stamps.

**Tree-Based Time-Stamping** Time-stamps can be stored in a hash tree rather than in a linked list. [1] This reduces storage requirements and time-stamp verification complexity.

The premise is that publicizing time-stamps, as required in linked time-stamping, is expensive and can only happen at longer time intervals. In order to be able to issue more than one time-stamp in one interval, a linked time-stamping

scheme may only publicize every  $N$ th time-stamp. A client then needs to keep up to  $N$  time-stamps to verify that it is part of the publicized chain.

In tree-based time-stamping, on the other hand, the hashes of  $N$  documents can be used to construct a hash tree, only the root of which is then publicized. A client then only needs to store  $\log N$  document hashes to verify that their document is part of the hash tree. While reducing storage requirements, tree-based time-stamping sacrifices the total order of time-stamps as it is provided by linked time-stamping.

**One-Way Accumulators** One-way accumulators are a "decentralized alternative to digital signatures" [2]. They can be used to construct a time-stamp protocol. [2]

A one-way accumulator is a one-way hash functions with a quasi-commutative property. The verification of a time-stamp based on one-way accumulators is even more efficient than in the tree-based approach: Only a constant amount of information is needed, regardless of the number of documents which were used to construct the publicized hash.

### **Combining Different Time-Stamping Schemes**

Linked and tree-based time-stamping both require that time-stamps be publicized in some widely accessible way (e. g. by printing them in newspapers). Bayer et al. [1] note that random-witness time-stamping not only achieves such publication among the witnesses, but with their signatures also provides a proof that the publication was witnessed. The authors conclude that the different approaches to digital time-stamping could be used in conjunction.

**Blockchains** The modern concept of a blockchain [5] is heavily inspired by Haber and Stornetta's approach to linked time-stamping. Blocks in the Bitcoin blockchain are time-stamps of financial transactions. These are chained together just as in linked time-stamping.

Bitcoin works in a permissionless setting by using a proof of work to determine which node is allowed to add a block to the chain, and by leveraging the longest chain rule to determine which of the potentially competing chains should be

trusted. The protocol achieves eventual consistency between the participating nodes without the need for a central authority.

### 3 Methodology and Research Questions

**RQ1** How can the time-stamp protocols existing in the literature be combined into a generalized framework for digital time-stamping?

In order to answer this research question, I will conduct a narrative literature review. I have already summarized what in my opinion seem to be the relevant time-stamp protocols in Section 2. I will start the work on my thesis by searching for any papers I have potentially missed up onto this point.

I will go on to identify common themes and building blocks shared among the different time-stamping schemes. I will try to combine these building blocks in a generalized time-stamping framework that, ideally, can be used to derive all of the existing time-stamp protocols and potentially even combine their building blocks in ways not seen in the literature before.

I will then construct hypotheses for potential trade-offs between different configurations of this generalized time-stamping framework. I will particularly focus on the trade-offs between temporal resolution, performance (time-stamp throughput and scalability) and security (against back-dating and denial of service).

RQ1 will lead to many new research questions concerning the trade-offs of time-stamp protocols. With RQ2, I focus on the questions directly related to the random-witness approach to keep the project within the scope of a thesis. I specifically choose the random-witness approach because of its interesting probabilistic properties. I have also already created a practical implementation of this protocol [3], which I can use to conduct experiments.

**RQ2** How does the choice of configuration parameters for the random-witness time-stamp algorithm affect temporal resolution, performance (throughput and scalability) and security

(against back-dating and denial of service)?

To answer this research question, I will first try to identify some hard theoretical trade-offs between temporal resolution, throughput and security. It seems a reasonable assumption that such trade-offs exist: For example, the achievable temporal resolution likely depends on the propagation delay of the network. Requiring a small temporal resolution may thus facilitate denial of service attacks which can increase propagation delays by congesting the network. As another example, the throughput of time-stamps that build a total order is obviously bounded by the inverse of the temporal resolution. However, the relationship between temporal resolution and throughput is more complex if time-stamps only form a partial order.

Based on the insights gained by theoretical considerations, I will construct hypotheses for the effect different configuration parameters should have on measurable quantities (like throughput, response time, bandwidth usage, temporal resolution of time-stamps, etc.).

I will finally set up experiments to test these hypotheses using my distributed time-stamping library written in Rust [3] and a network emulation software like Mininet or Toxiproxy. Emulation allows me to easily change between different network configurations and run experiments at a low cost.

### 4 Timeline

I aim to achieve the following milestones by the specified times.

- **March 20:** Finish the narrative literature review, combining the different existing time-stamp protocols into one generalized framework for time-stamping. Discuss trade-offs between different configurations within this time-stamping framework. Choose a network emulation tool and get to know it.
- **April 17:** Identify the potential theoretical limits of and trade-offs between time-stamp throughput, temporal resolution and accuracy and write the corresponding sections in

the thesis. Construct hypotheses and set up experiments for network emulation.

- **May 15:** Run and evaluate the experiments, write a first draft of the entire thesis.

As explained in Section 3, I expect the thesis to open up many more research questions concerning the trade-offs of time-stamp protocols. If I can complete the timeline above quicker than expected and have additional time on my hands, I will start tackling these questions.

## References

- [1] Dave Bayer, Stuart Haber, and W. Scott Stornetta. Improving the efficiency and reliability of digital time-stamping. In Renato Capocelli, Alfredo De Santis, and Ugo Vaccaro, editors, *Sequences II*, pages 329–334, New York, NY, 1993. Springer New York.
- [2] Josh Benaloh and Michael de Mare. One-Way Accumulators: A Decentralized Alternative to Digital Signatures. In Tor Helleseth, editor, *Advances in Cryptology — EUROCRYPT ’93*, pages 274–285, Berlin, Heidelberg, 1994. Springer.
- [3] Keno Goertz. dts. <https://gitlab.informatik.hu-berlin.de/goertzke/dts>. Accessed: 2025-02-20.
- [4] Stuart Haber and W. Scott Stornetta. How to time-stamp a digital document. *Journal of Cryptology*, 3(2):99–111, January 1991.
- [5] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Satoshi Nakamoto*, 2008.
- [6] Daniela Tulone. A Scalable and Intrusion-tolerant Digital Time-stamping System. In *2006 IEEE International Conference on Communications*, volume 5, pages 2357–2363, June 2006. ISSN: 1938-1883.